Lecture 20

Plan:

1) Finish LCIS algo.
2) Matroid intersection polytope

Matroid intersection polytope

- Let

- analogously to the matroid polytope, let

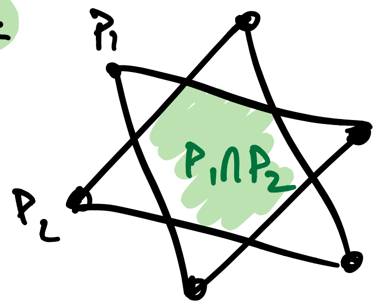
$$X = \{ \qquad \qquad \}$$

ie.

- Define the matroid intersection polytope
- Main result:

- This is surprising!

e.g.



- In terms of inequalities?
- Recall matroid polytope:

$P_M = \{ \qquad \qquad \qquad \}.$

- $P_{M_1} \cap P_{M_2}$ has

Theorem: Let $P = P_{M_1} \cap P_{M_2}$, i.e.

$P = \{ x \in \mathbb{R}^E :$

$\}$

Then

Proof: Plan:

- Like second proof for matroid polytope,
- Integrality suffices by the usual logic:

▷

D

- Again,

- But

<u>Let x^* be an extreme point of P.</u>

- We know

- For $i \in \{1,2\}$, let

$$T_i = \left\{ \qquad\qquad\qquad \right\}$$

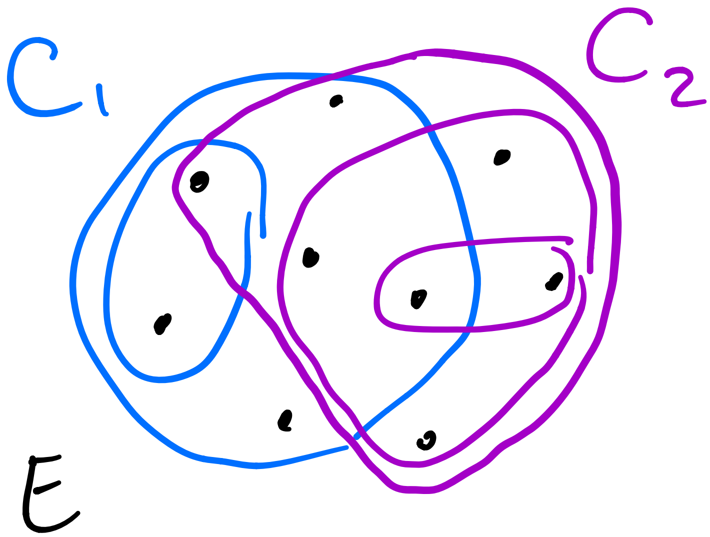
- Let $J = \{ \qquad\qquad \}$.

- Then x^* is

- That is, $\{x^*\}$ is

- Recall from lec 17:

e.g.



- Thus, assume

-

Claim: $\Rightarrow$

- Why?

e.g.

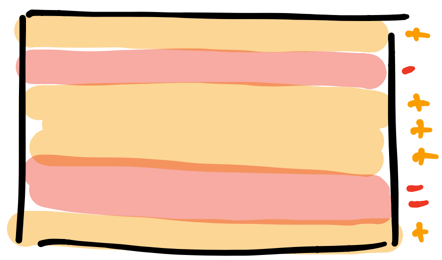
$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

- We use

- **Recall**: A T.U. $\Leftrightarrow$ $\forall$ submatrices A' of A

e.g.

A'



$= \quad 1 \; 0 \; 0 \; -1 \; 1 \; \ldots \; 1$

- Consider submatrix A' of A

- Assign

e.g.

▷ For

e.g.

- Overall,

$\square$.

Matroid intersection optimization

- Given a cost function

- For just one matroid:

- For $c = 1$:

- For perfect matching:

- can also compute

Exercise:

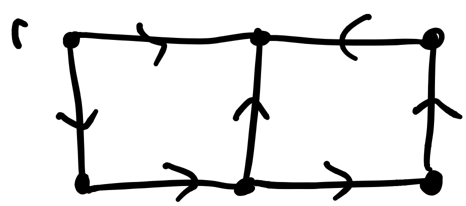
- In general,

- Today:

Min-cost arborescence

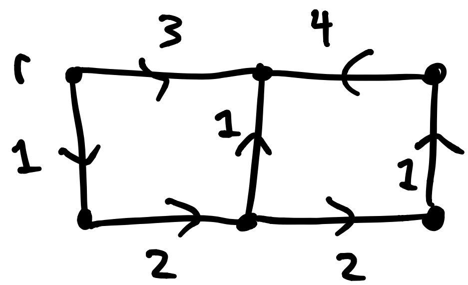
- **Recall:** given directed graph D & vertex r, **arborescence** A

e.g.



- min-cost arborescence:

e.g.



- e.g. edges =

  r =

  Cost =

- First, I.P. formulation:

assume

$$OPT = \min_{x \in \mathbb{R}^E}$$

subject to

- **Check**: only solutions are

- **Miraculously**, we'll show

- I.e. the following L.P. has

LP = min

subject to

(primal)

( ).

- Dual LP is

LP = max

subject to

(dual)

- **Algorithm sketch**: construct
  - ▷
  - ▷

Then

- Complementary slackness

  a.)

  b.)

- Two phases of algorithm:

  1) Construct

     ▷

     ▷

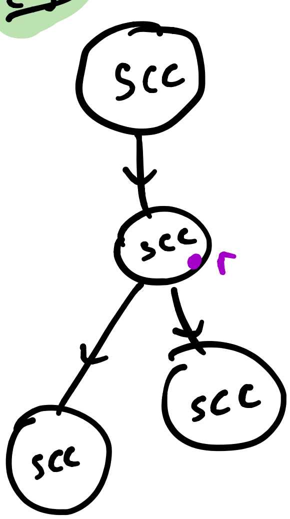
2) Remove unnecessary edges

Phase 1 Initialize

- While

  - select

e.g.



i)

ii)

▷ increase

(

).

▷

▷ <u>Return</u>

Phase 2: eliminate as many edges as we can

- For
  - If
- Return

Claim 1:

Pf: . We'll show

- If
- if

• suppose

$\square$

finally:

Claim 2:

a.)

Pf: Assume not

•

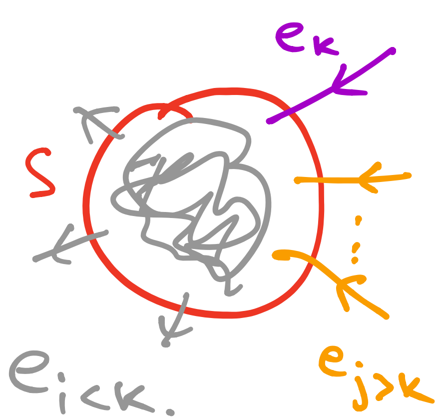
•

( )

$\Rightarrow$

•

$\Rightarrow$

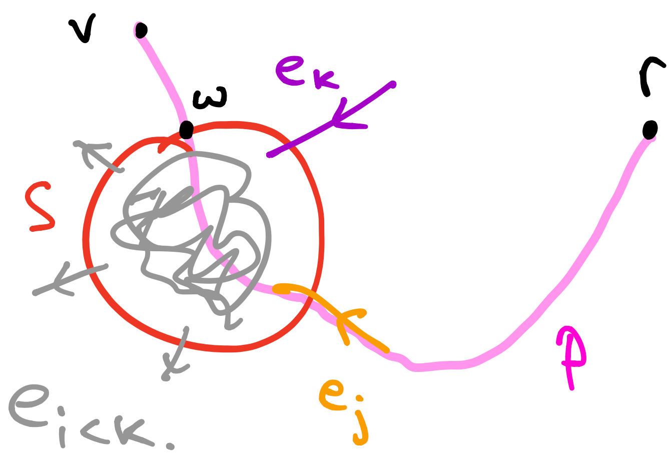


• **Subclaim:**

Why?

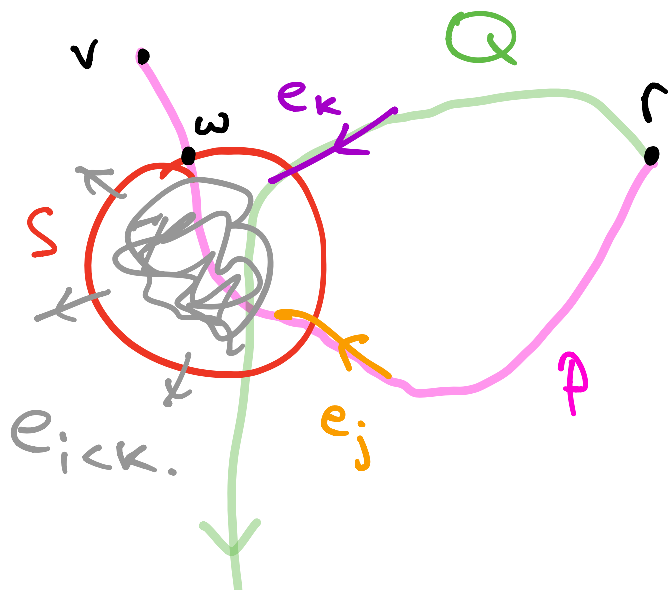
• Let

• Let



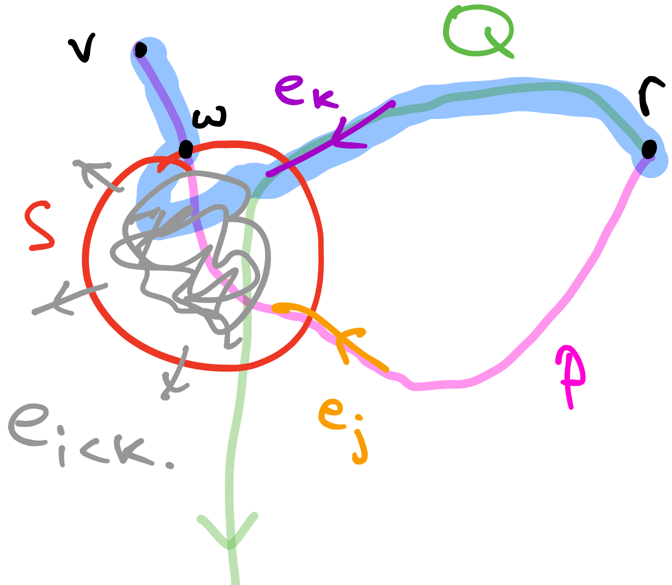
Note:

• Because



similarly:

• Can



$\square$.